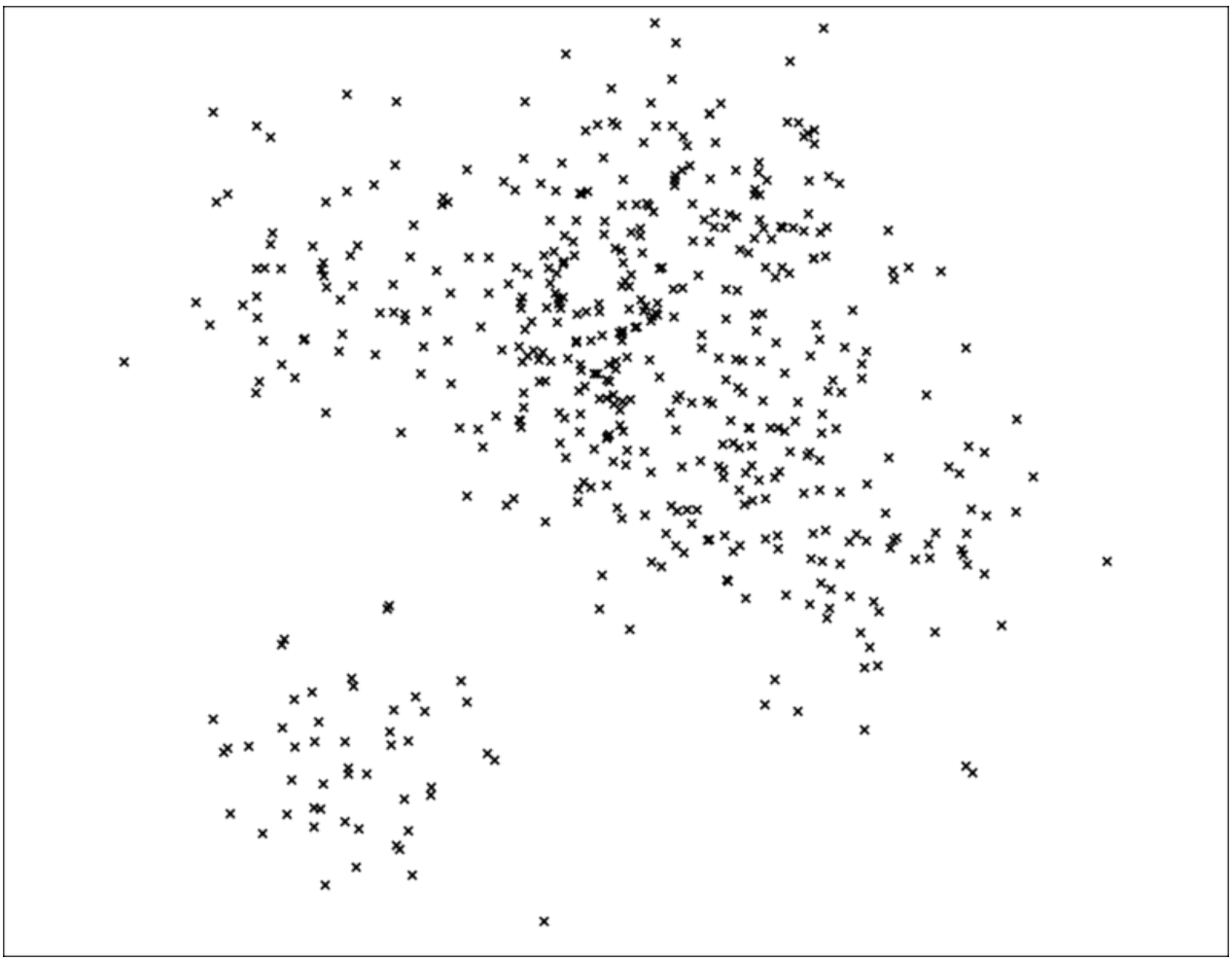
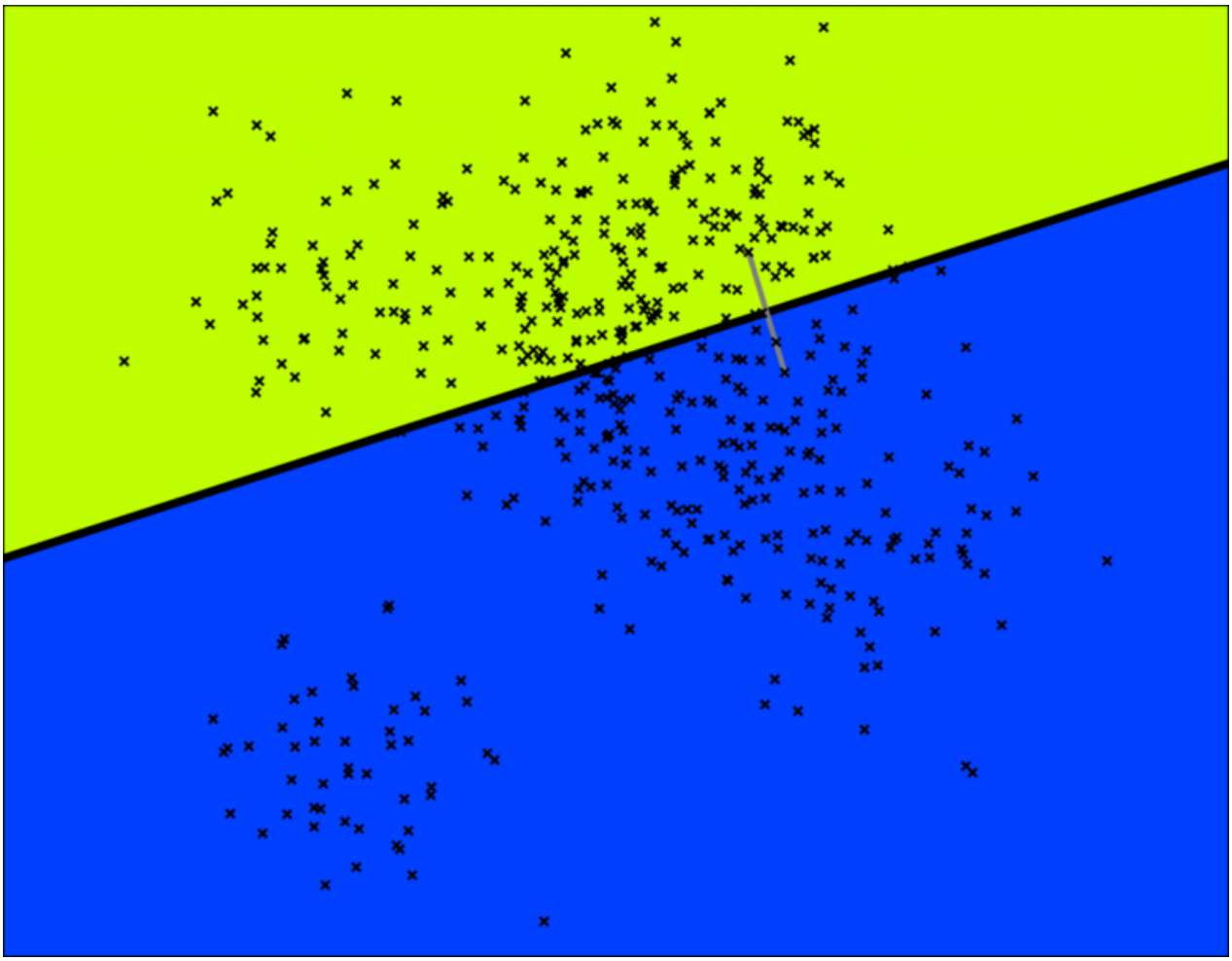
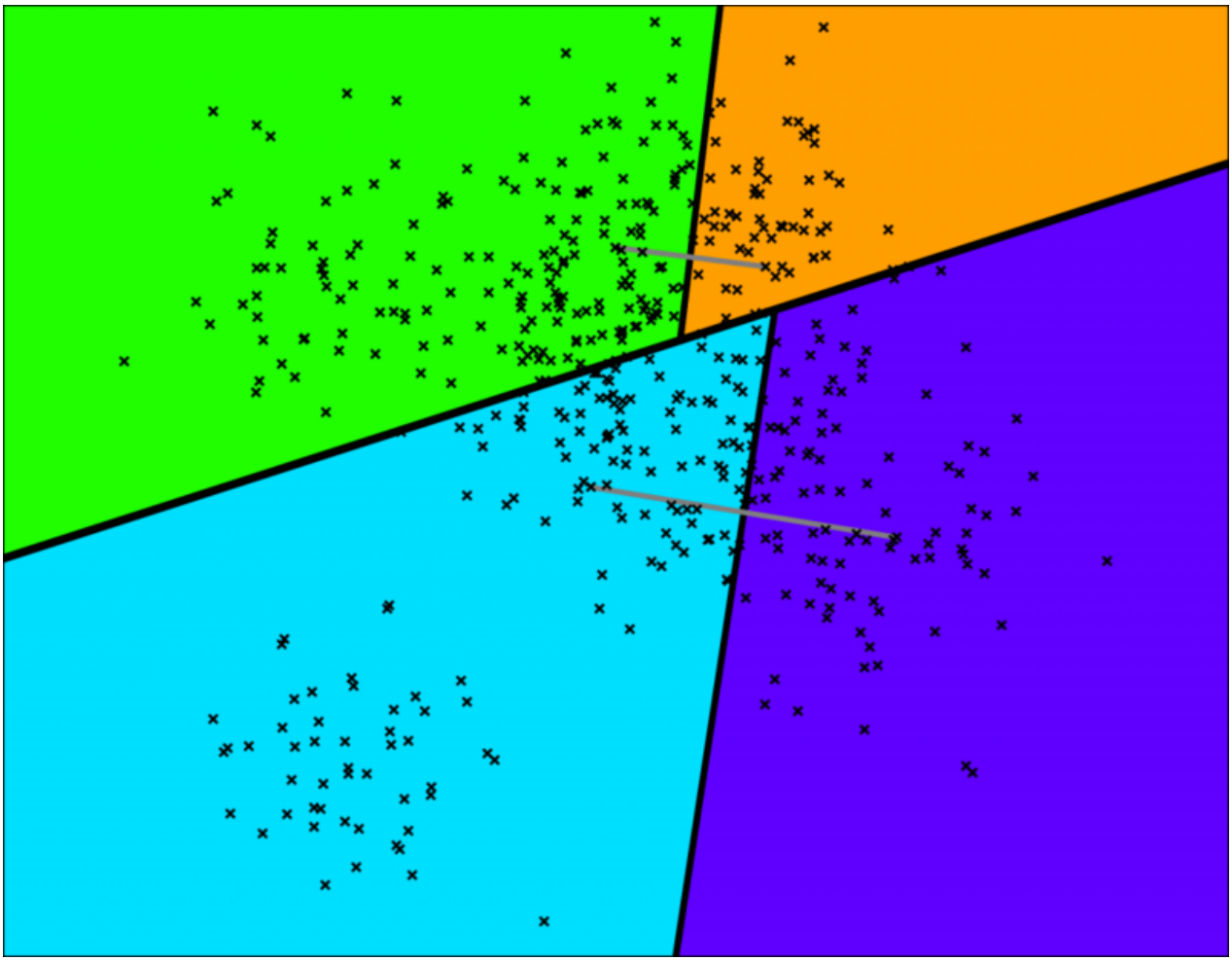
Annoy搜索算法(Approximate Nearest Neighbors Oh Yeah



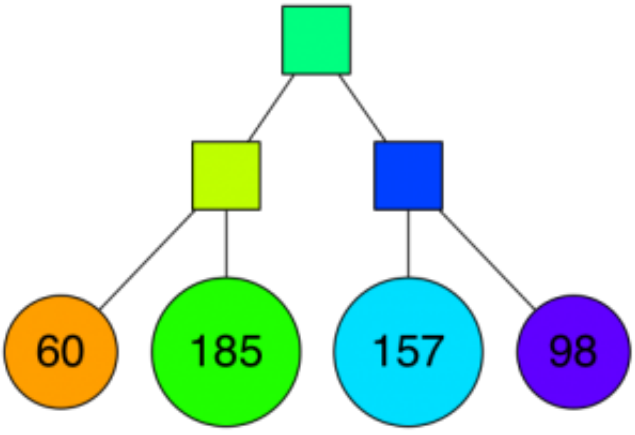
annoy 算法的目标是建立一个数据结构能够在较短的时间内找到任何查询点的最近点，在精度允许的条件下通过牺牲准确率来换取比暴力搜索要快的多的搜索速度。



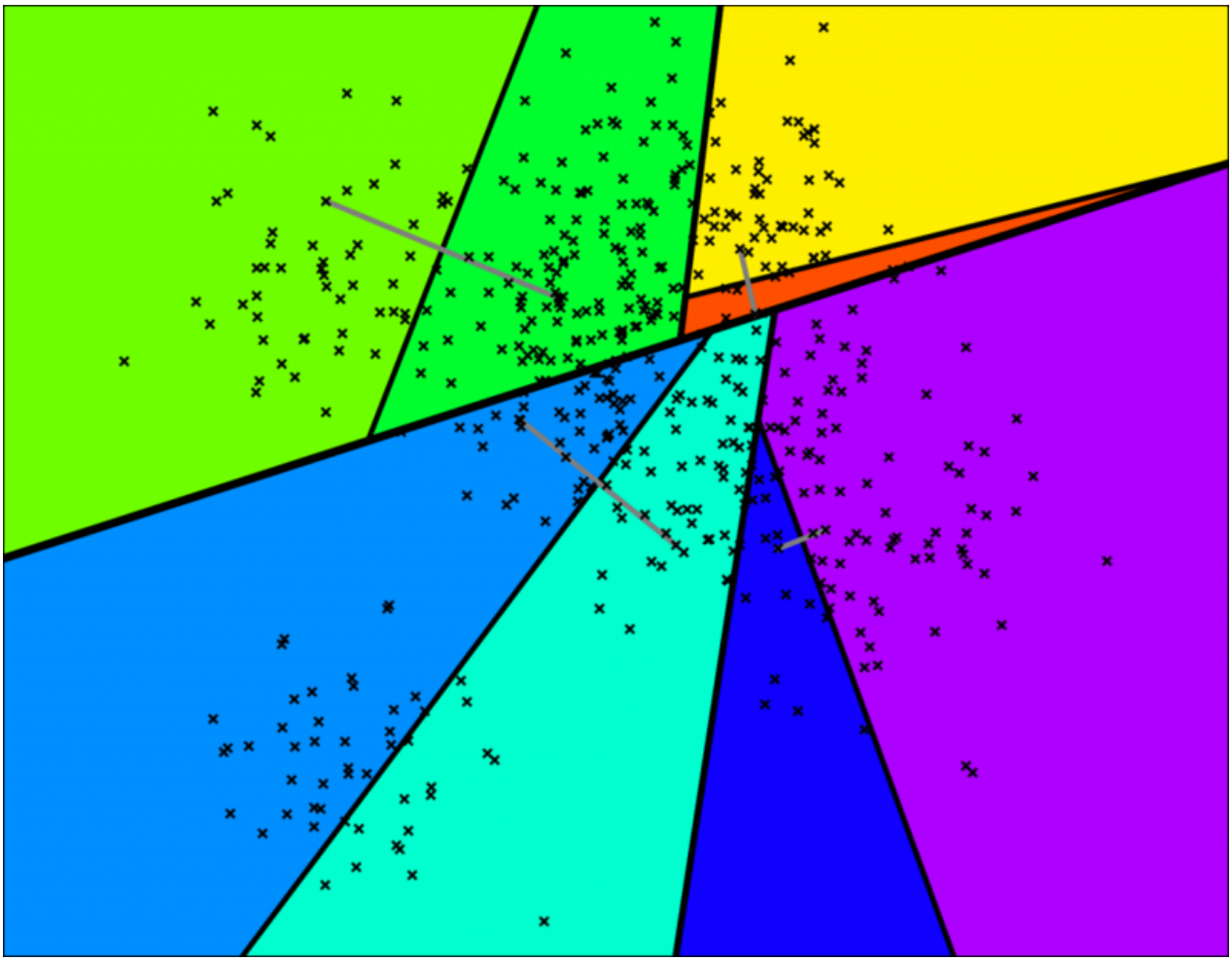
首先随机选择两个点，然后根据这两个点之间的连线确定一个可以垂直等分线段的超平面，灰色是两点的连线，黑色是超平面。



接下里在超平面分割后的字空间内按照同样的方法继续确定超平面分割字空间，通过这样的方法我们可以将子空间的从属关系用二叉树来表示：

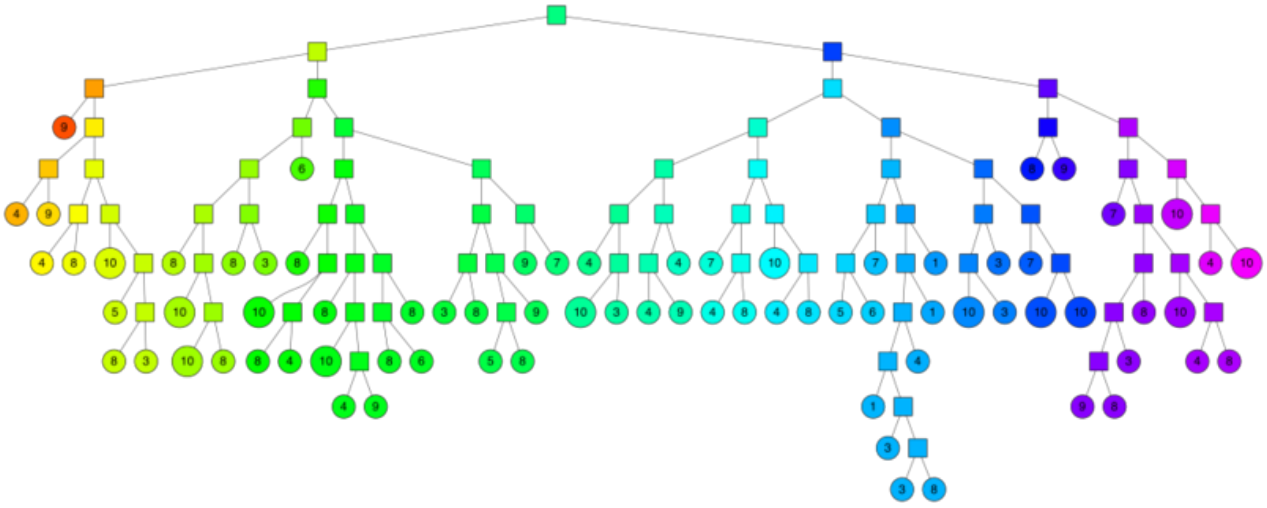


然后再继续分割：

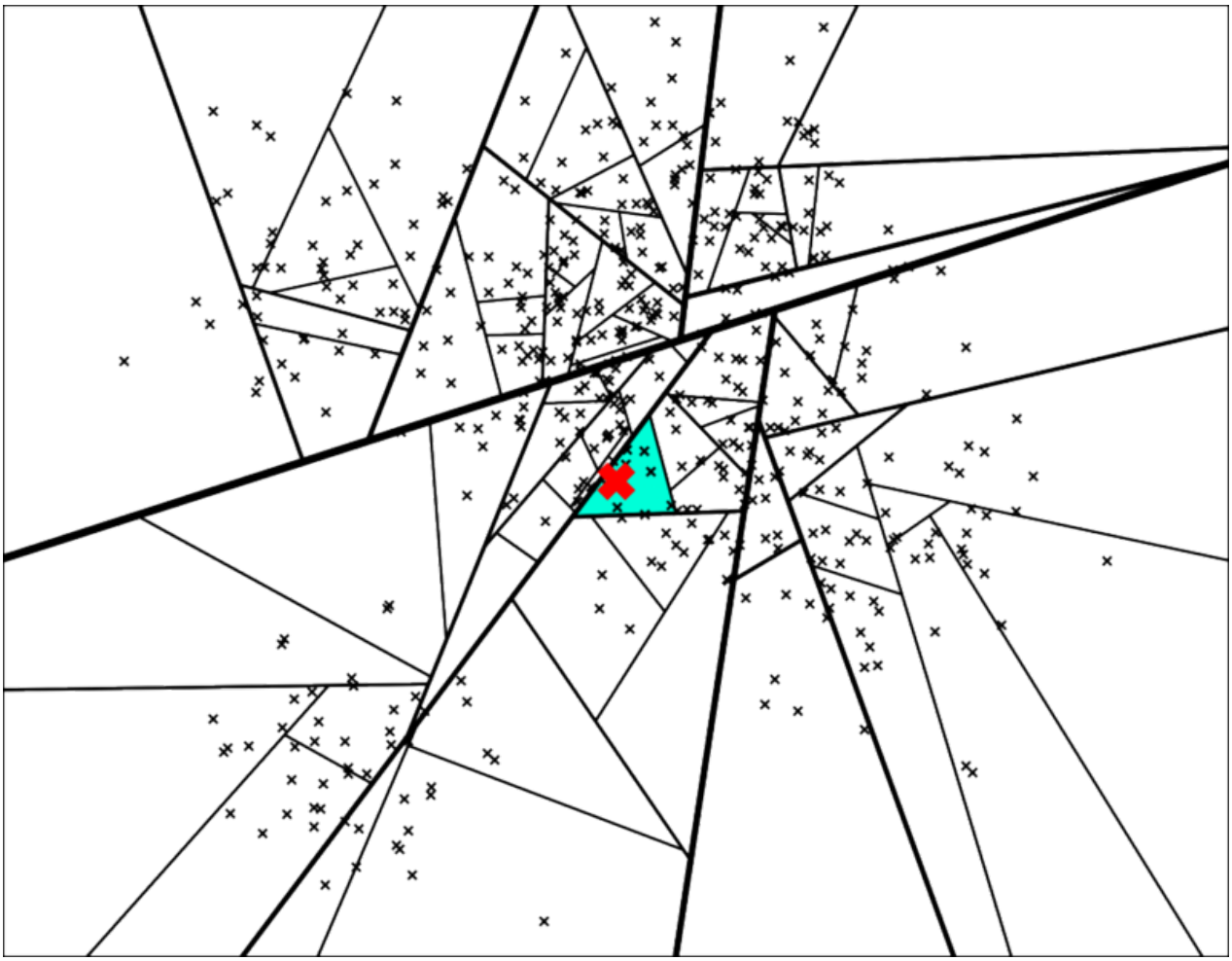


继续重复上述步骤，直到子节点包含的数据点数不超过 K 个，这里我们取 K = 10。

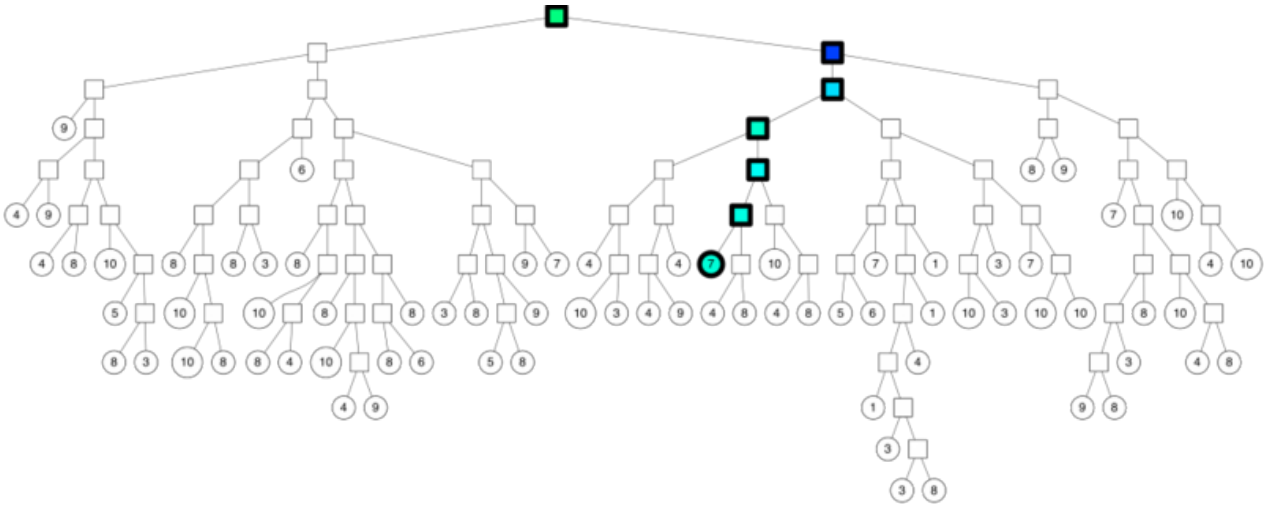
对应的二叉树结构如下所示：



通过上述步骤，我们建立了二叉树的结构用于表示上述点分布空间，每个节点都表示一个子空间，在点分布空间中接近的子空间在二叉树结构中表现为位置靠近的节点。这里有一个假设，如果两个点在空间中彼此靠近，任何超平面都不可能将他们分开。 如果要搜索空间中的任意一个点，我们都可以从根结点遍历二叉树。 假设我们要找下图中红色 X 表示的点的临近点：



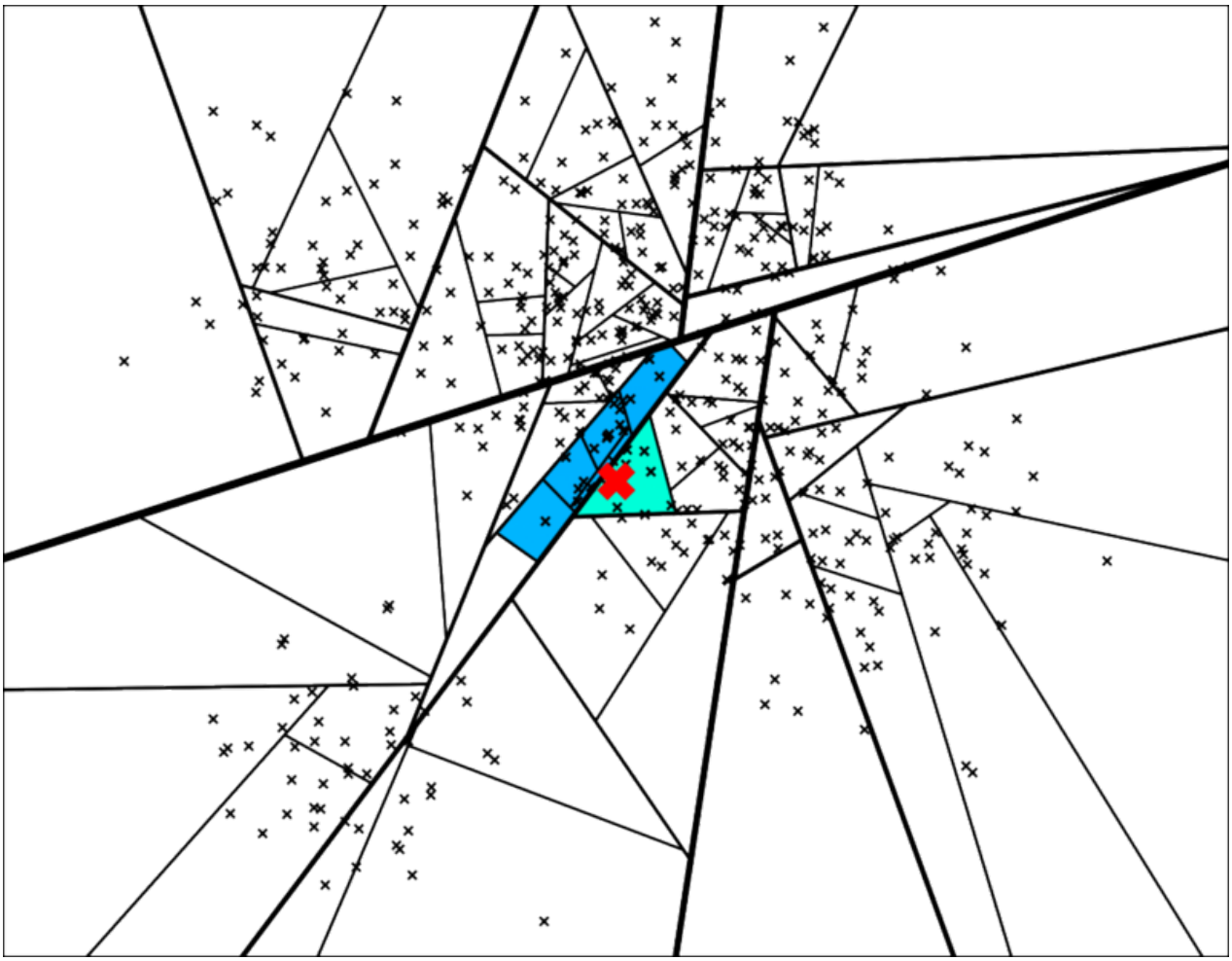
从二叉树的根结点遍历的路径如下所示：



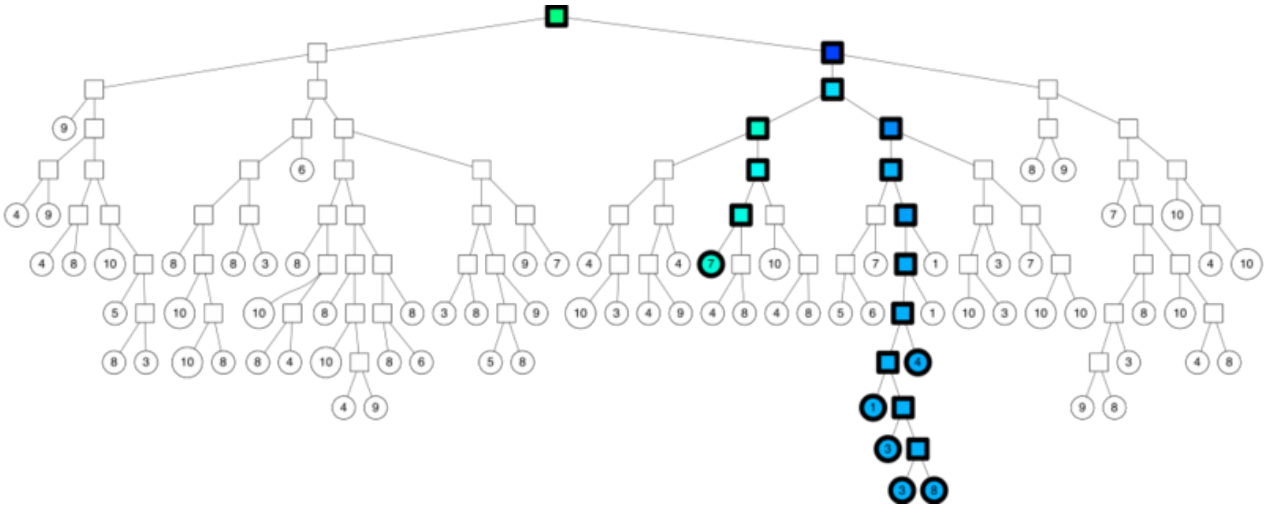
可以看到在最后的子节点中有7个点，其中有6个是我们要找的临近点。不过这样找到的点太少了，我们可以采用其他办法。

使用优先队列

不仅是在二叉树中找红色点存在的路径，还要找与红色点相近的子空间存在的路径（最大路径为子空间与红色点的最远距离）。

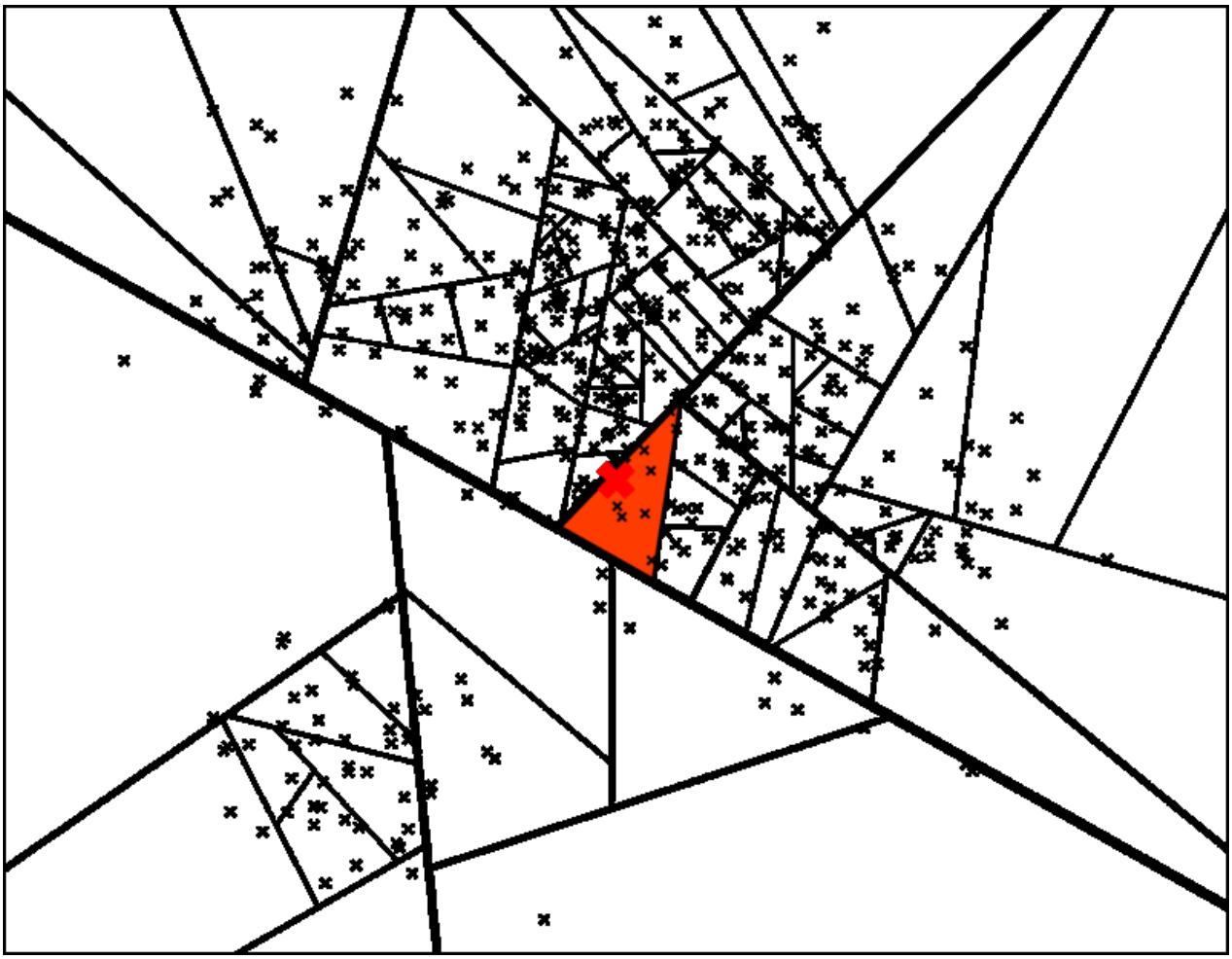


对应的二叉树结构如下所示：

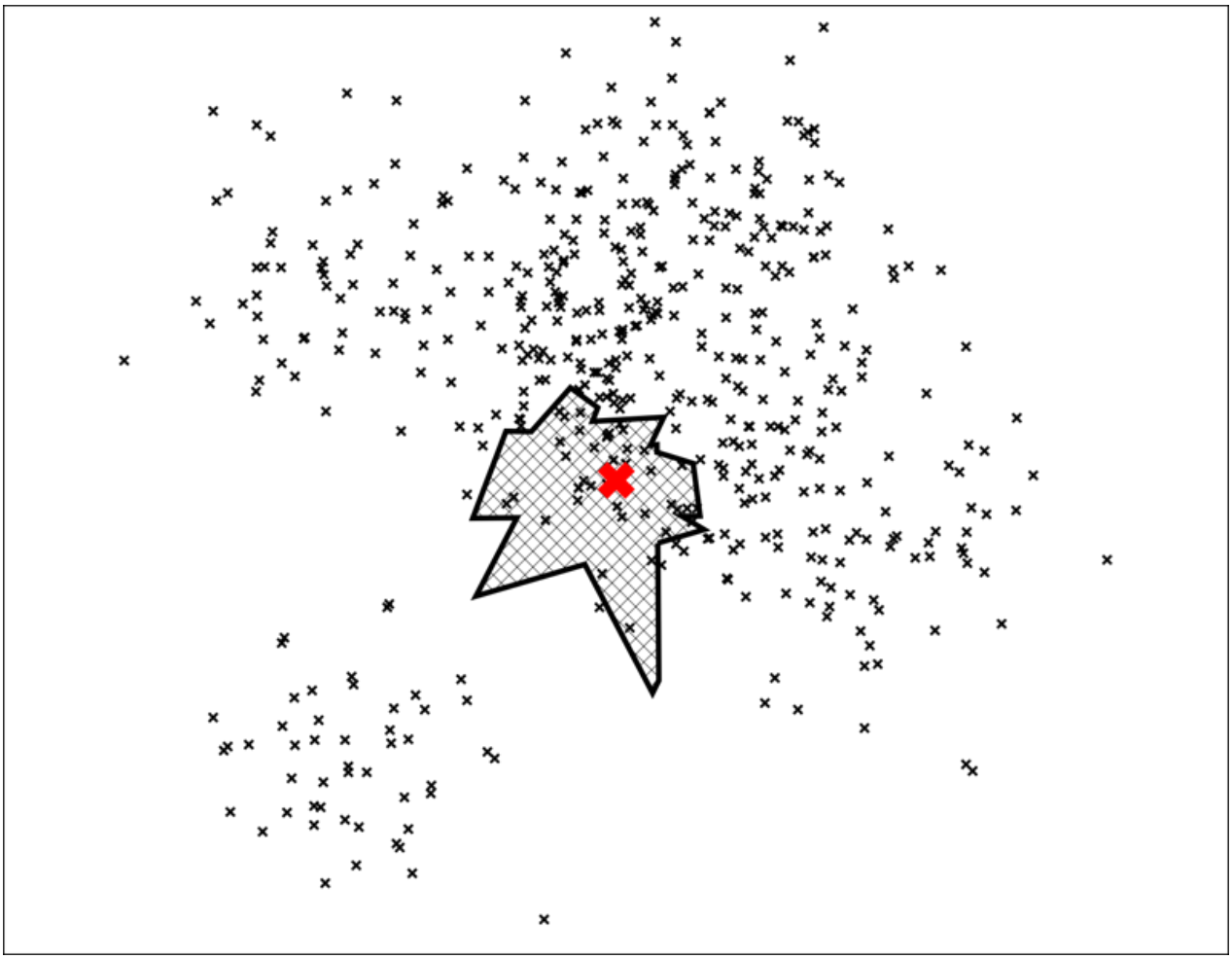


建立多个二叉树

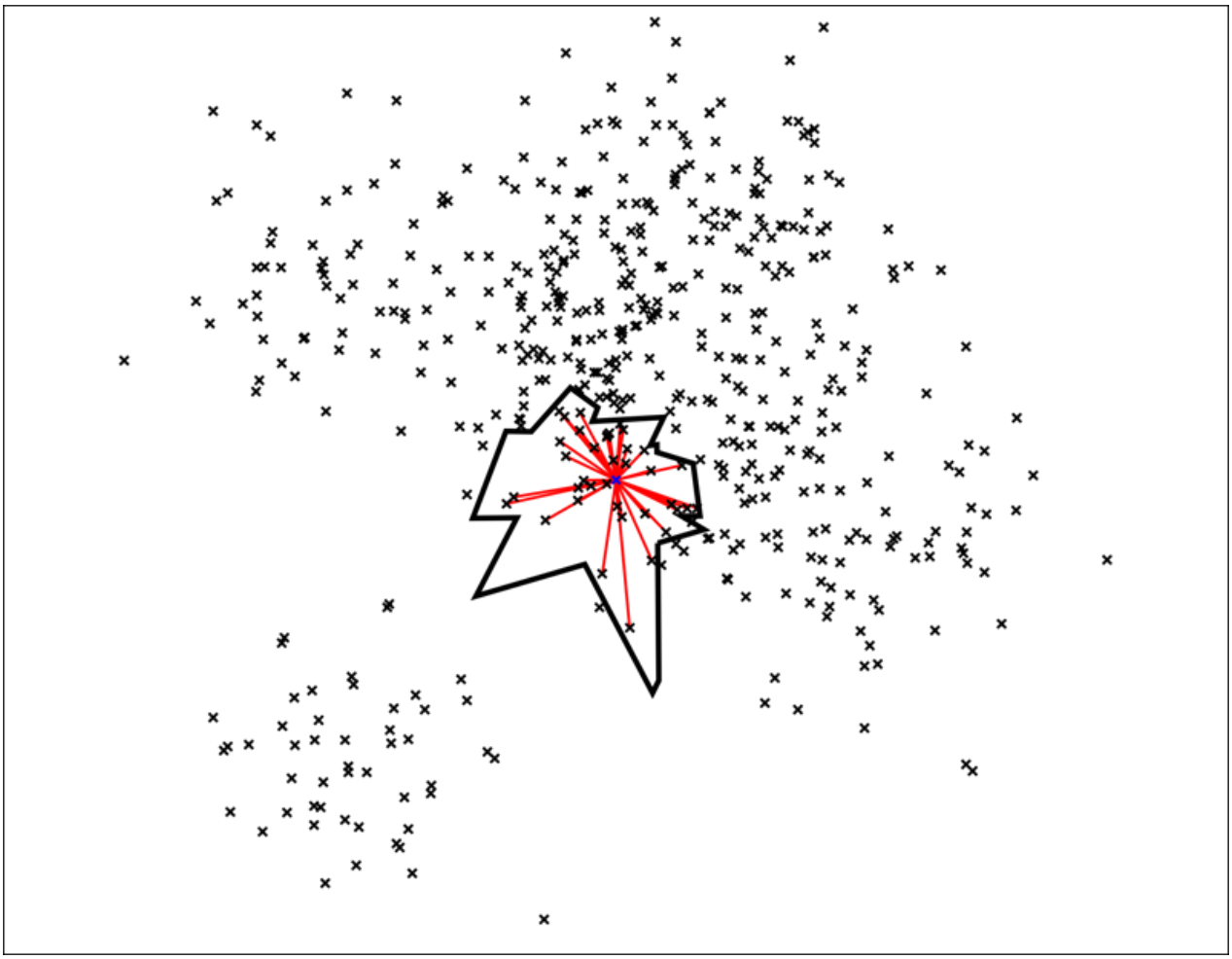
第二个技巧就是构建多个二叉树结构构成森林，每棵树都是通过随机选择点分割平面构建的，我们需要同时在所有树结构中寻找相邻点：



每棵树都包含平面中所有点，所以当我们搜索多棵树的时候，将这些节点组合起来，我们可以得到对于目标点的近邻点分布的大概估计：



接下来再对这个分布范围内所有点进行距离计算并排序：



排序后我们再选择最前面的 K 个点，这就是 Annoy 算法的工作原理。当然会有一部分本应该是相邻的点没有包含进来，不过损失了一定精度的情况下可以换来比穷举搜索快很多的搜索时间。